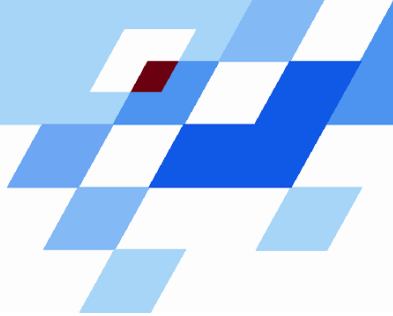


Wintersemester 2006/07

Einführung in die Informatik für Naturwissenschaftler und Ingenieure

(alias **Einführung in die Programmierung)**

(Vorlesung)

Prof. Dr. Günter Rudolph

Fachbereich Informatik

Lehrstuhl für Algorithm Engineering

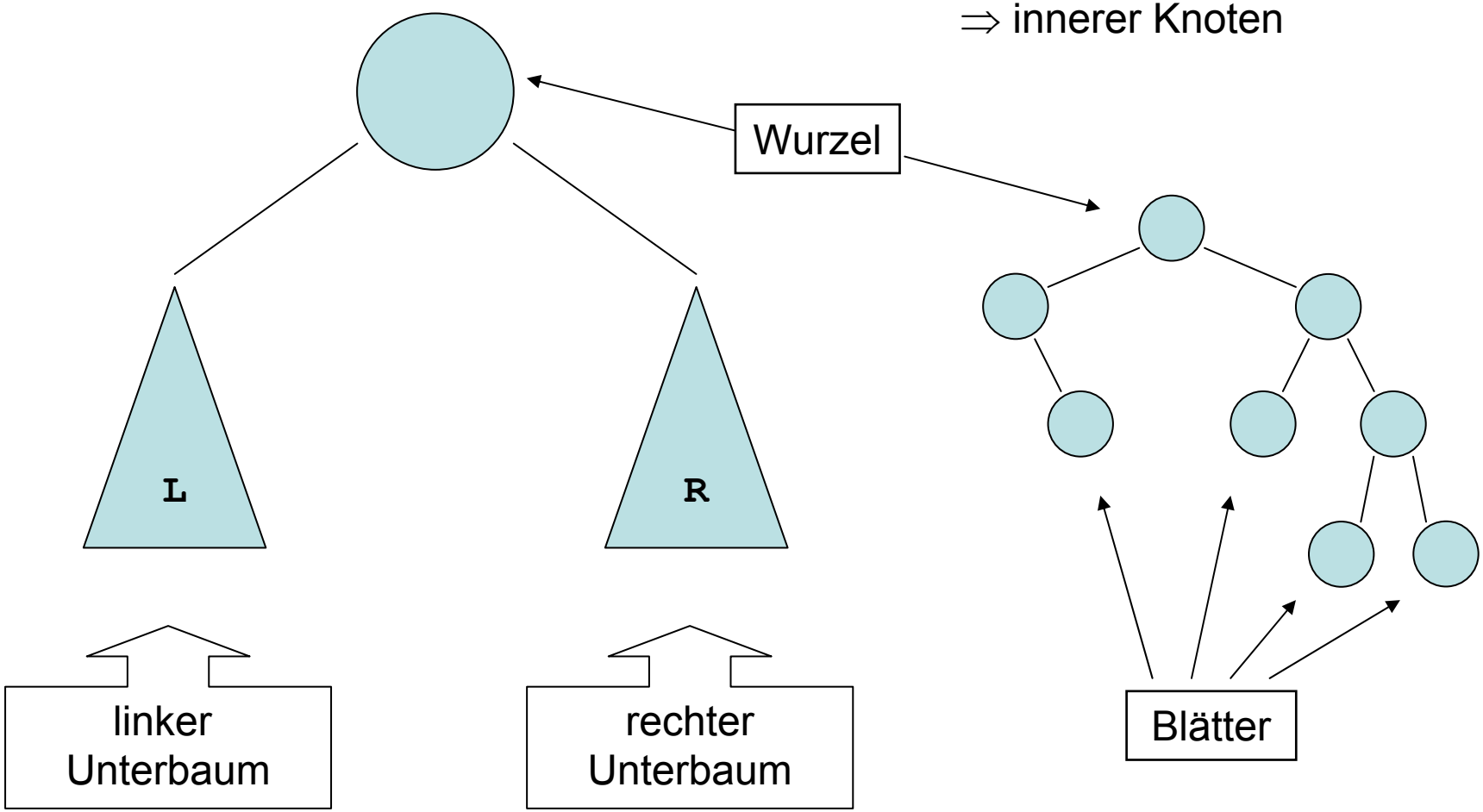
Inhalt

- Definition ☑
- ADT Keller ☑
- ADT Schlange ☑
  - Exkurs: Dynamischer Speicher (new / delete)
- ADT Liste ☑
- ADT Binärbaum ☑
  - Exkurs: Einfache Dateibehandlung
  - Exkurs: Strings (C++) … endlich! ☺

ADT Binäre Bäume: Terminologie

keine Wurzel und kein Blatt

⇒ innerer Knoten



ADT Binäre Bäume: Datenstruktur

```
struct BinTree {
  T data;                    // Nutzdaten
  BinTree *lTree, *rTree;    // linker und rechter Unterbaum
};
```

Falls ein Unterbaum nicht existiert, dann zeigt der Zeiger auf 0.

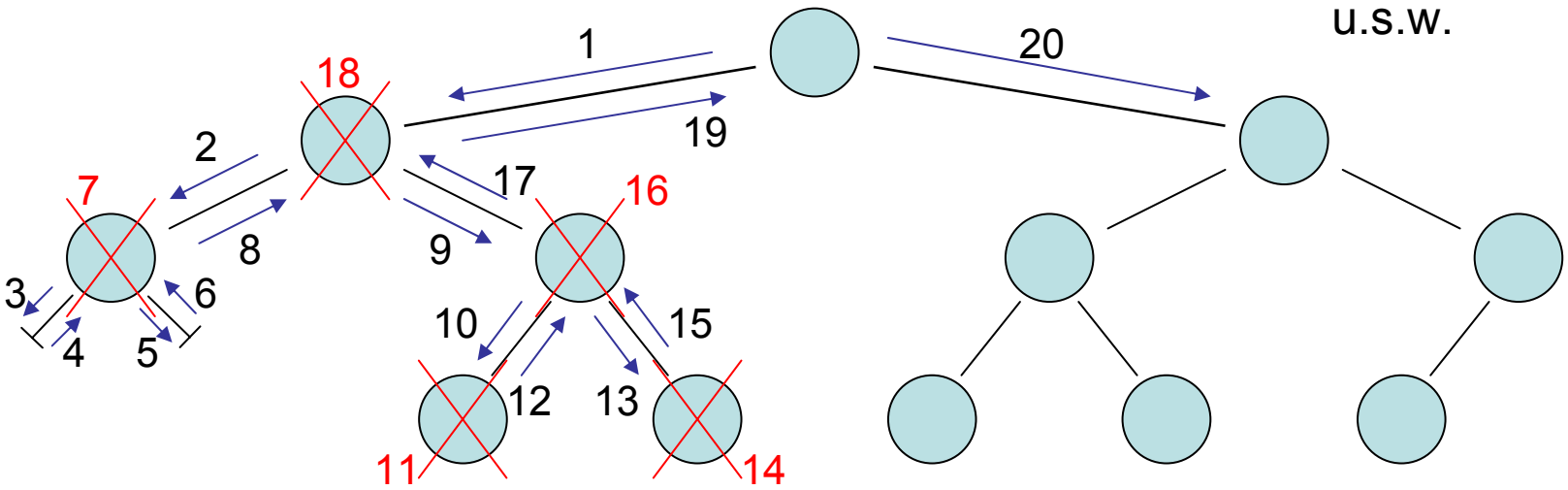
```
bool IsElement(int key, BinTree *tree) {
  if (tree == 0) return false;
  if (tree->data == key) return true;
  if (tree->data < key) return IsElement(key, tree->rTree);
  return IsElement(key, tree->lTree);
}
```

ADT Binäre Bäume: Einfügen

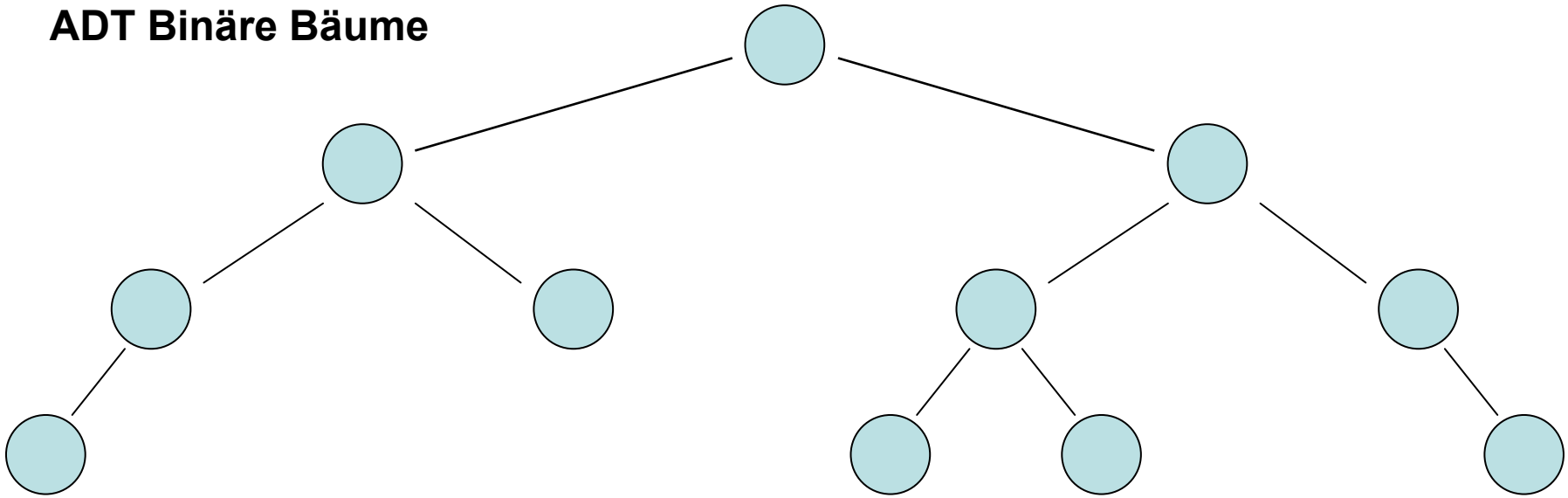
```
BinTree *Insert(int key, BinTree *tree) {
  if (tree == 0) {
    BinTree *b = new BinTree;
    b->data = key;
    b->lTree = b->rTree = 0;
    return b;
  }
  else {
    if (tree->data < key)
      tree->rTree = Insert(key, tree->rTree);
    else if (tree->data > key)
      tree->lTree = Insert(key, tree->lTree);
    return tree;
  }
}
```

ADT Binäre Bäume: Aufräumen

```
void Clear(BinTree *tree) {
  if (tree == 0) return;    // Rekursionsabbruch
  Clear(tree->lTree);       // linken Unterbaum löschen
  Clear(tree->rTree);       // rechten Unterbaum löschen
  delete tree;              // aktuellen Knoten löschen
}
```



ADT Binäre Bäume



Höhe := Länge des längsten Pfades von der Wurzel zu einem Blatt.

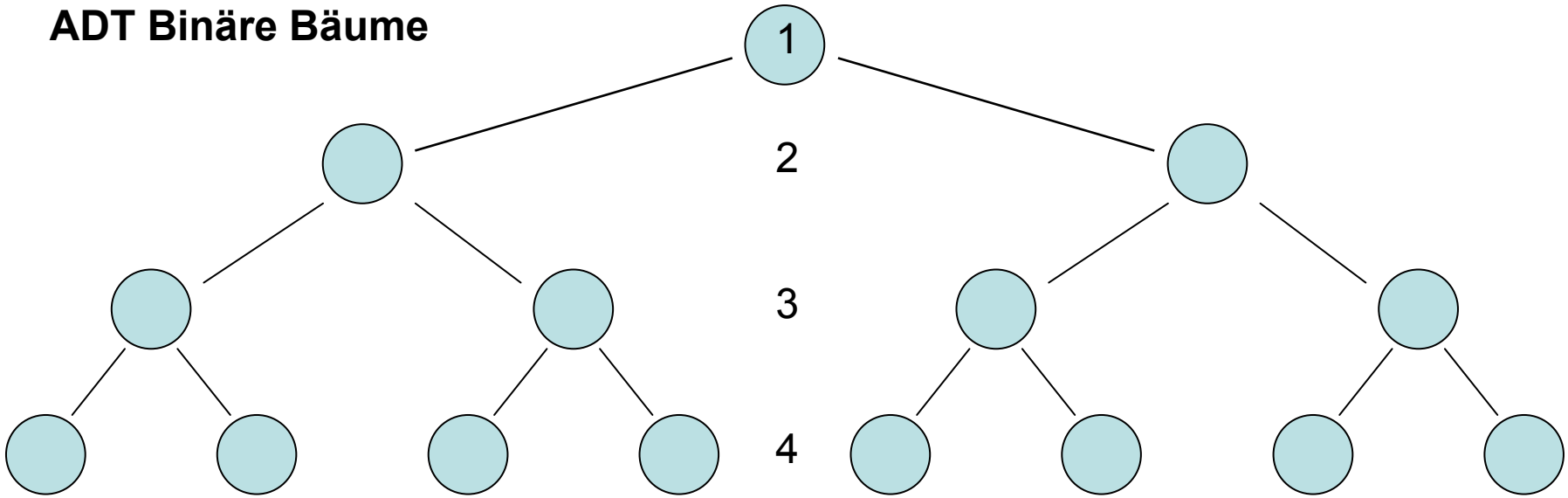
Höhe(leerer Baum) = 0

Höhe(nicht leerer Baum) = 1 + max { Höhe(linker U-Baum), Höhe(rechter U-Baum) }

Anmerkung: rekursive Definition!

(U-Baum = Unterbaum)

ADT Binäre Bäume

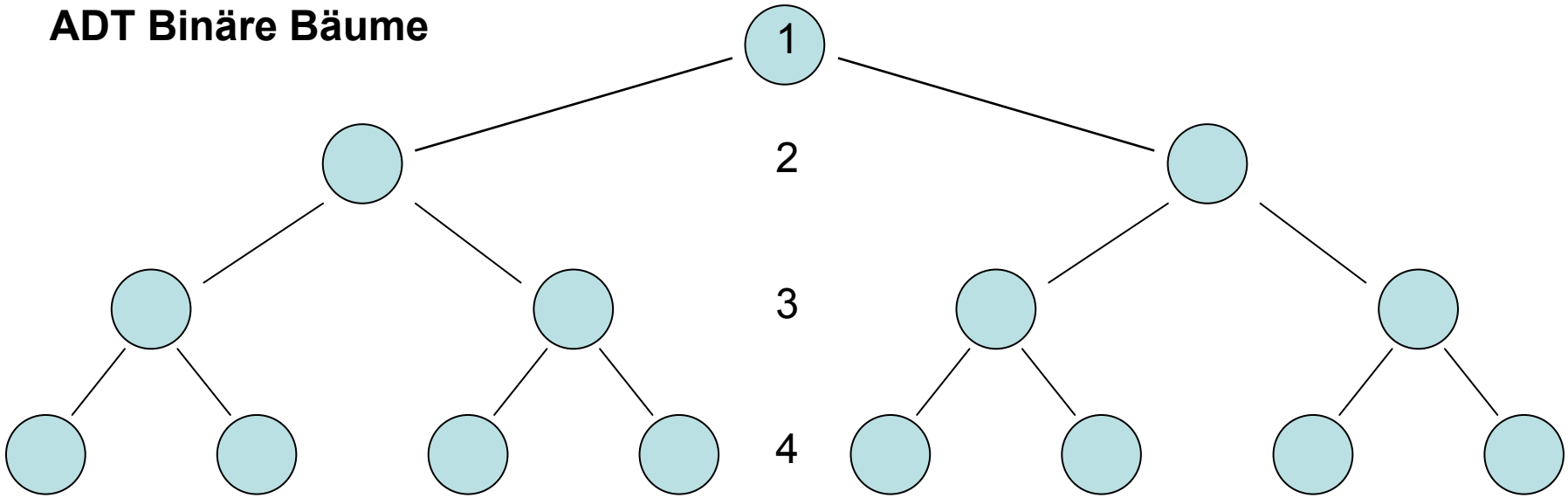


Auf Ebene k können jeweils zwischen 1 und 2^{k-1} Elemente gespeichert werden.

$\Rightarrow$ In einem Baum der Höhe h können also zwischen h und

$$\sum_{k=1}^{h} 2^{k-1} = 2^h - 1$$ Elemente gespeichert werden!

ADT Binäre Bäume



- Ein **vollständiger Baum** der Höhe h besitzt $2^h - 1$ Knoten.
  Man braucht maximal h Vergleiche, um Element (ggf. nicht) zu finden.
  Bei $n = 2^h - 1$ Elementen braucht man $\log_2(n) < h$ Vergleiche!

- Ein **degenerierter Baum** der Höhe h besitzt h Knoten (= lineare Liste).
  Man braucht maximal h Vergleiche, um Element (ggf. nicht) zu finden.
  Bei n = h braucht man also n Vergleiche!

Exkurs: Einfache Dateibehandlung

Datei := speichert Daten in linearer Anordnung

Zwei Typen:

- **ASCII-Dateien**
  - sind mit Editor les- und schreibbar
  - Dateiendung („suffix“ oder „extension“) meist `.txt` oder `.asc`
  - betriebssystem-spezifische Übersetzung von Zeichen bei Datentransfer zwischen Programm und externem Speicher
- **Binär-Dateien**
  - werden byteweise beschrieben und gelesen
  - lesen / schreiben mit Editor ist keine gute Idee
  - schnellerer Datentransfer, da keine Zeichenübersetzung

Exkurs: Einfache Dateibehandlung

Hier: einfache Dateibehandlung!

- Dateien können **gelesen** oder **beschrieben** werden.
- Vor dem ersten Lesen oder Schreiben muss **Datei geöffnet** werden.
- Man kann prüfen, ob das Öffnen funktioniert hat.
- Nach dem letzten Lesen oder Schreiben muss **Datei geschlossen** werden.
- Bei zu lesenden Dateien kann gefragt werden, ob **Ende der Datei** erreicht ist.
- Beim Öffnen einer zu schreibenden Datei wird vorheriger Inhalt gelöscht!
- Man kann noch viel mehr machen …

wir benötigen:

```
#include <fstream>      bzw.      <fstream.h>
```

Exkurs: Einfache Dateibehandlung

- Eingabe-Datei = **i**nput **f**ile

```
ifstream Quelldatei;
```

- Ausgabe-Datei = **o**utput **f**ile

```
ofstream Zieldatei;
```

- Öffnen der Datei:

```
Quelldatei.open(dateiName);
```

ist Kurzform von
Quelldatei.open(dateiName, modus);

wobei fehlender modus bedeutet:
ASCII-Datei,
Eingabedatei (weil ifstream)

- Öffnen der Datei:

```
Zieldatei.open(dateiName);
```

ist Kurzform von
Quelldatei.open(dateiName, modus);

wobei fehlender modus bedeutet:
ASCII-Datei,
Ausgabedatei (weil ofstream)

Exkurs: Einfache Dateibehandlung

modus:

`ios::binary`	binäre Datei
`ios::in`	öffnet für Eingabe (implizit bei ifstream)
`ios::out`	öffnet für Ausgabe (implizit bei ofstream)
`ios::app`	hängt Daten am Dateiende an
`ios::nocreate`	wenn Datei existiert, dann nicht anlegen

Warnung: teilweise Compiler-abhängig
(`nocreate` fehlt in MS VS 2003, dafür `trunc`)

Man kann diese Schalter / Flags miteinander kombinieren via:

`ios::binary | ios::app` (öffnet als binäre Datei und hängt Daten an)

Exkurs: Einfache Dateibehandlung

- **Datei öffnen**
  `file.open(fileName)` bzw. `file.open(fileName, modus)`
  falls Öffnen fehlschlägt, wird Nullpointer zurückgegeben
- **Datei schließen**
  `file.close()`
  sorgt für definierten Zustand der Datei auf Dateisystem;
  bei nicht geschlossenen Dateien droht Datenverlust!
- **Ende erreicht?**
  ja falls `file.eof() == true`
- **Lesen** (von ifstream)
  `file.get(c);` liest ein Zeichen
  `file >> x;` liest verschiedene Typen
- **Schreiben** (von ofstream)
  `file.put(c);` schreibt ein Zeichen
  `file << x;` schreibt verschiedene Typen

Exkurs: Einfache Dateibehandlung

Merke:

1. Auf eine geöffnete Datei darf immer nur einer zugreifen.
2. Eine geöffnete Datei belegt Ressourcen des Betriebssystems.
   $\Rightarrow$ Deshalb Datei nicht länger als nötig geöffnet halten.
3. Eine geöffnete Datei unbekannter Länge kann solange gelesen werden, bis das Ende-Bit (end of file, EOF) gesetzt wird.
4. Der Versuch, eine nicht vorhandene Datei zu öffnen (zum Lesen) oder eine schreibgeschützte Datei zu öffnen (zum Schreiben), führt zu einem Nullpointer in ifstream bzw. ofstream.
   $\Rightarrow$ Das muss überprüft werden, sonst Absturz bei weiterer Verwendung!
5. Dateieingabe und -ausgabe (input/output, I/O) ist sehr langsam im Vergleich zu den Rechenoperationen.
   $\Rightarrow$ I/O Operationen minimieren.

"The fastest I/O is no I/O."

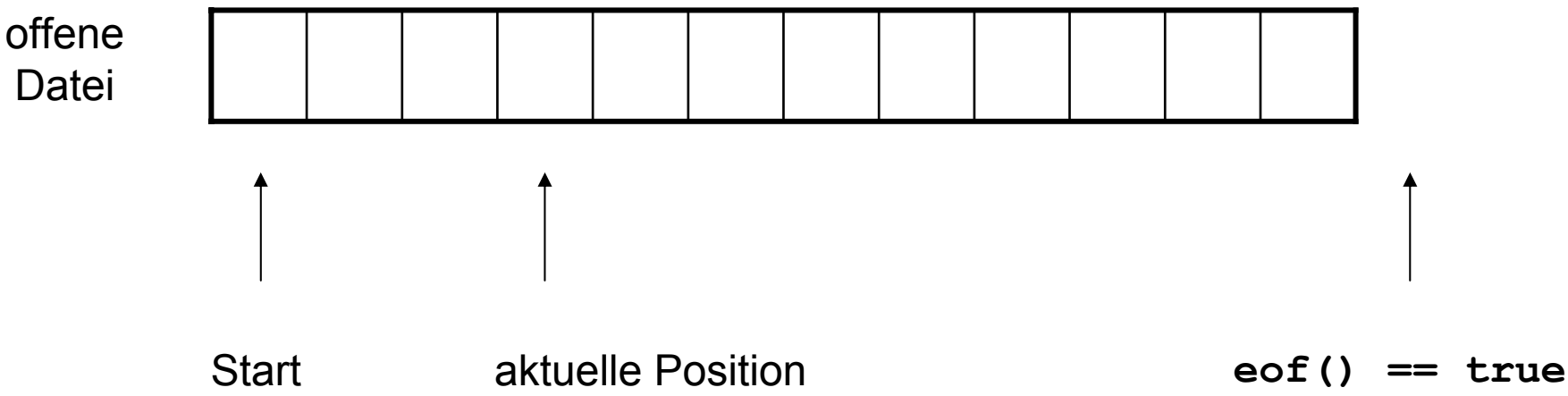
Nils-Peter Nelson, Bell Labs

Exkurs: Einfache Dateibehandlung

```
#include <iostream>
#include <fstream>

using namespace std;

int main() {                    // zeichenweise kopieren
  ifstream Quelldatei;
  ofstream Zieldatei;

  Quelldatei.open("quelle.txt");
  if (!Quelldatei) {
    cerr << "konnte Datei nicht zum Lesen öffnen\n";
    exit(1);
  }
  Zieldatei.open("ziel.txt");
  if (!Zieldatei) {
    cerr << "konnte Datei nicht zum Schreiben öffnen\n";
    exit(1);
  }
```

Exkurs: Einfache Dateibehandlung

```
  while (!Quelldatei.eof()) {
    char c;
    Quelldatei.get(c);
    Zieldatei.put(c);
  }

  Quelldatei.close();
  Zieldatei.close();
}
```



Exkurs: C++ Strings

Bisher:

Zeichenketten wie `char str[20];`

→ Relikt aus C-Programmierung!

→ bei größeren Programmen mühevoll, lästig, …

→ … und insgesamt **fehlerträchtig!**

Jetzt:

Zeichenketten aus C++

→ sehr angenehm zu verwenden (keine 0 am Ende, variable Größe, …)

→ eingebaute (umfangreiche) Funktionalität

wie benötigen: `#include <string>` und `using namespace std;`

Datendefinition / Initialisierung

```
string s1;                  // leerer String
string s2 = "xyz";          // initialisieren mit C-String
string s3 = s2;             // vollständige Kopie!
string s4("abc");           // initialisieren mit C-String
string s5(s4);              // initialisieren mit C++-String
string s6(10, '*');         // ergibt String aus 10 mal *
string s7(1,'x');           // initialisieren mit einem char
string sx('x');             // FEHLER!
string s8("");              // leerer String
```

Eingebaute Funktionen

- Konvertierung C++-String nach C-String via `c_str()`

```
const char *Cstr = s2.c_str();
```

- Stringlänge `length()`

```
cout << s2.length();
```

- Index von Teilstring finden

```
int pos = s2.find("yz");
```

- Strings addieren

```
s1 = s2 + s3;
s4 = s2 + "hello";
s5 += s4;
```

- Strings vergleichen

```
if (s1 == s2) s3 += s2;
if (s3 < s8) flag = true;
```

- `substr()`, `replace()`, `erase()`, ...